NAME

postgres - PostgreSQL database server

SYNOPSIS

postgres [*option*...]

DESCRIPTION

postgres is the PostgreSQL database server. In order for a client application to access a database it connects (over a network or locally) to a running **postgres** instance. The **postgres** instance then starts a separate server process to handle the connection.

One **postgres** instance always manages the data of exactly one database cluster. A database cluster is a collection of databases that is stored at a common file system location (the "data area"). More than one **postgres** instance can run on a system at one time, so long as they use different data areas and different communication ports (see below). When **postgres** starts it needs to know the location of the data area. The location must be specified by the **-D** option or the **PGDATA** environment variable; there is no default. Typically, **-D** or **PGDATA** points directly to the data area directory created by **initdb**(1). Other possible file layouts are discussed in Section 20.2.

By default **postgres** starts in the foreground and prints log messages to the standard error stream. In practical applications **postgres** should be started as a background process, perhaps at boot time.

The **postgres** command can also be called in single-user mode. The primary use for this mode is during bootstrapping by **initdb**(1). Sometimes it is used for debugging or disaster recovery; note that running a single-user server is not truly suitable for debugging the server, since no realistic interprocess communication and locking will happen. When invoked in single-user mode from the shell, the user can enter queries and the results will be printed to the screen, but in a form that is more useful for developers than end users. In the single-user mode, the session user will be set to the user with ID 1, and implicit superuser powers are granted to this user. This user does not actually have to exist, so the single-user mode can be used to manually recover from certain kinds of accidental damage to the system catalogs.

OPTIONS

postgres accepts the following command-line arguments. For a detailed discussion of the options consult Chapter 20. You can save typing most of these options by setting up a configuration file. Some (safe) options can also be set from the connecting client in an application-dependent way to apply only for that session. For example, if the environment variable **PGOPTIONS** is set, then libpq-based clients will pass that string to the server, which will interpret it as **postgres** command-line options.

General Purpose

-B *nbuffers*

Sets the number of shared buffers for use by the server processes. The default value of this parameter is chosen automatically by initdb. Specifying this option is equivalent to setting the shared_buffers configuration parameter.

-c *name=value*

Sets a named run-time parameter. The configuration parameters supported by PostgreSQL are described in Chapter 20. Most of the other command line options are in fact short forms of such a parameter assignment. **-c** can appear multiple times to set multiple parameters.

-C *name*

Prints the value of the named run-time parameter, and exits. (See the **-c** option above for details.) This returns values from postgresql.conf, modified by any parameters supplied in this invocation. It does not reflect parameters supplied when the cluster was started.

This can be used on a running server for most parameters. However, the server must be shut down for some runtime-computed parameters (e.g., shared_memory_size, shared_memory_size_in_huge_pages, and wal_segment_size).

This option is meant for other programs that interact with a server instance, such as **pg_ctl**(1), to query configuration parameter values. User-facing applications should instead use **SHOW** or the pg_settings view.

-d *debug-level*

Sets the debug level. The higher this value is set, the more debugging output is written to the server log. Values are from 1 to 5. It is also possible to pass -d 0 for a specific session, which will prevent the server log level of the parent **postgres** process from being propagated to this session.

-D *datadir*

Specifies the file system location of the database configuration files. See Section 20.2 for details.

-e

Sets the default date style to "European", that is DMY ordering of input date fields. This also causes the day to be printed before the month in certain date output formats. See Section 8.5 for more information.

-F

Disables **fsync** calls for improved performance, at the risk of data corruption in the event of a system crash. Specifying this option is equivalent to disabling the fsync configuration parameter. Read the detailed documentation before using this!

-h *hostname*

Specifies the IP host name or address on which **postgres** is to listen for TCP/IP connections from client applications. The value can also be a comma-separated list of addresses, or * to specify listening on all available interfaces. An empty value specifies not listening on any IP addresses, in which case only Unix-domain sockets can be used to connect to the server. Defaults to listening only on localhost. Specifying this option is equivalent to setting the listen_addresses configuration parameter.

-i

Allows remote clients to connect via TCP/IP (Internet domain) connections. Without this option, only local connections are accepted. This option is equivalent to setting *listen_addresses* to * in postgresql.conf or via **-h**.

This option is deprecated since it does not allow access to the full functionality of listen_addresses. It's usually better to set *listen_addresses* directly.

-k *directory*

Specifies the directory of the Unix-domain socket on which **postgres** is to listen for connections from client applications. The value can also be a comma-separated list of directories. An empty value specifies not listening on any Unix-domain sockets, in which case only TCP/IP sockets can be used to connect to the server. The default value is normally /tmp, but that can be changed at build time. Specifying this option is equivalent to setting the unix_socket_directories configuration parameter.

-l

Enables secure connections using SSL. PostgreSQL must have been compiled with support for SSL for this option to be available. For more information on using SSL, refer to Section 19.9.

-N *max-connections*

Sets the maximum number of client connections that this server will accept. The default value of this parameter is chosen automatically by initdb. Specifying this option is equivalent to setting the max_connections configuration parameter.

-p *port*

Specifies the TCP/IP port or local Unix domain socket file extension on which **postgres** is to listen for connections from client applications. Defaults to the value of the **PGPORT** environment variable, or if **PGPORT** is not set, then defaults to the value established during compilation (normally 5432). If you specify a port other than the default port, then all client applications must specify the same port using either command-line options or **PGPORT**.

-s

Print time information and other statistics at the end of each command. This is useful for benchmarking or for use in tuning the number of buffers.

-S *work-mem*

Specifies the base amount of memory to be used by sorts and hash tables before resorting to temporary disk files. See the description of the *work_mem* configuration parameter in Section 20.4.1.

-V

--version

Print the postgres version and exit.

--*name***=***value*

Sets a named run-time parameter; a shorter form of **-c**.

--describe-config

This option dumps out the server's internal configuration variables, descriptions, and defaults in tab-delimited **COPY** format. It is designed primarily for use by administration tools.

-?

--help

Show help about postgres command line arguments, and exit.

Semi-Internal Options

The options described here are used mainly for debugging purposes, and in some cases to assist with recovery of severely damaged databases. There should be no reason to use them in a production database setup. They are listed here only for use by PostgreSQL system developers. Furthermore, these options might change or be removed in a future release without notice.

-f { s | i | o | b | t | n | m | h }

Forbids the use of particular scan and join methods: s and i disable sequential and index scans respectively, o, b and t disable index-only scans, bitmap index scans, and TID scans respectively, while n, m, and h disable nested-loop, merge and hash joins respectively.

Neither sequential scans nor nested-loop joins can be disabled completely; the -fs and -fn options simply discourage the optimizer from using those plan types if it has any other alternative.

-n

This option is for debugging problems that cause a server process to die abnormally. The ordinary

strategy in this situation is to notify all other server processes that they must terminate and then reinitialize the shared memory and semaphores. This is because an errant server process could have corrupted some shared state before terminating. This option specifies that **postgres** will not reinitialize shared data structures. A knowledgeable system programmer can then use a debugger to examine shared memory and semaphore state.

-O

Allows the structure of system tables to be modified. This is used by **initdb**.

-P

Ignore system indexes when reading system tables, but still update the indexes when modifying the tables. This is useful when recovering from damaged system indexes.

-t pa[rser] | pl[anner] | e[xecutor]

Print timing statistics for each query relating to each of the major system modules. This option cannot be used together with the **-s** option.

-T

This option is for debugging problems that cause a server process to die abnormally. The ordinary strategy in this situation is to notify all other server processes that they must terminate and then reinitialize the shared memory and semaphores. This is because an errant server process could have corrupted some shared state before terminating. This option specifies that **postgres** will stop all other server processes by sending the signal SIGSTOP, but will not cause them to terminate. This permits system programmers to collect core dumps from all server processes by hand.

-v *protocol*

Specifies the version number of the frontend/backend protocol to be used for a particular session. This option is for internal use only.

-W *seconds*

A delay of this many seconds occurs when a new server process is started, after it conducts the authentication procedure. This is intended to give an opportunity to attach to the server process with a debugger.

Options for Single-User Mode

The following options only apply to the single-user mode (see Single-User Mode below).

--single

Selects the single-user mode. This must be the first argument on the command line.

database

Specifies the name of the database to be accessed. This must be the last argument on the command line. If it is omitted it defaults to the user name.

-E

Echo all commands to standard output before executing them.

-j

Use semicolon followed by two newlines, rather than just newline, as the command entry terminator.

-r *filename*

Send all server log output to *filename*. This option is only honored when supplied as a command-line option.

ENVIRONMENT

PGCLIENTENCODING

Default character encoding used by clients. (The clients can override this individually.) This value can also be set in the configuration file.

PGDATA

Default data directory location

PGDATESTYLE

Default value of the DateStyle run-time parameter. (The use of this environment variable is deprecated.)

PGPORT

Default port number (preferably set in the configuration file)

DIAGNOSTICS

A failure message mentioning semget or shmget probably indicates you need to configure your kernel to provide adequate shared memory and semaphores. For more discussion see Section 19.4. You might be able to postpone reconfiguring your kernel by decreasing shared_buffers to reduce the shared memory consumption of PostgreSQL, and/or by reducing max_connections to reduce the semaphore consumption.

A failure message suggesting that another server is already running should be checked carefully, for example by using the command

$ **ps ax | grep postgres**

or

$ **ps -ef | grep postgres**

depending on your system. If you are certain that no conflicting server is running, you can remove the lock file mentioned in the message and try again.

A failure message indicating inability to bind to a port might indicate that that port is already in use by some non-PostgreSQL process. You might also get this error if you terminate **postgres** and immediately restart it using the same port; in this case, you must simply wait a few seconds until the operating system closes the port before trying again. Finally, you might get this error if you specify a port number that your operating system considers to be reserved. For example, many versions of Unix consider port numbers under 1024 to be "trusted" and only permit the Unix superuser to access them.

NOTES

The utility command **pg_ctl**(1) can be used to start and shut down the **postgres** server safely and comfortably.

If at all possible, *do not* use SIGKILL to kill the main **postgres** server. Doing so will prevent **postgres** from freeing the system resources (e.g., shared memory and semaphores) that it holds before terminating. This might cause problems for starting a fresh **postgres** run.

To terminate the **postgres** server normally, the signals SIGTERM, SIGINT, or SIGQUIT can be used. The first will wait for all clients to terminate before quitting, the second will forcefully disconnect all clients, and the third will quit immediately without proper shutdown, resulting in a recovery run during restart.

The SIGHUP signal will reload the server configuration files. It is also possible to send SIGHUP to an individual server process, but that is usually not sensible.

To cancel a running query, send the SIGINT signal to the process running that command. To terminate a backend process cleanly, send SIGTERM to that process. See also **pg_cancel_backend** and **pg_terminate_backend** in Section 9.27.2 for the SQL-callable equivalents of these two actions.

The **postgres** server uses SIGQUIT to tell subordinate server processes to terminate without normal cleanup. This signal *should not* be used by users. It is also unwise to send SIGKILL to a server process -- the main **postgres** process will interpret this as a crash and will force all the sibling processes to quit as part of its standard crash-recovery procedure.

BUGS

The -- options will not work on FreeBSD or OpenBSD. Use **-c** instead. This is a bug in the affected operating systems; a future release of PostgreSQL will provide a workaround if this is not fixed.

SINGLE-USER MODE

To start a single-user mode server, use a command like

postgres --single -D /usr/local/pgsql/data *other-options* **my_database**

Provide the correct path to the database directory with **-D**, or make sure that the environment variable **PGDATA** is set. Also specify the name of the particular database you want to work in.

Normally, the single-user mode server treats newline as the command entry terminator; there is no intelligence about semicolons, as there is in psql. To continue a command across multiple lines, you must type backslash just before each newline except the last one. The backslash and adjacent newline are both dropped from the input command. Note that this will happen even when within a string literal or comment.

But if you use the **-j** command line switch, a single newline does not terminate command entry; instead, the sequence semicolon-newline-newline does. That is, type a semicolon immediately followed by a completely empty line. Backslash-newline is not treated specially in this mode. Again, there is no intelligence about such a sequence appearing within a string literal or comment.

In either input mode, if you type a semicolon that is not just before or part of a command entry terminator, it is considered a command separator. When you do type a command entry terminator, the multiple statements you've entered will be executed as a single transaction.

To quit the session, type EOF (Control+D, usually). If you've entered any text since the last command entry terminator, then EOF will be taken as a command entry terminator, and another EOF will be needed to exit.

Note that the single-user mode server does not provide sophisticated line-editing features (no command history, for example). Single-user mode also does not do any background processing, such as automatic checkpoints or replication.

EXAMPLES

To start **postgres** in the background using default values, type:

$ **nohup postgres >logfile 2>&1 </dev/null &**

To start **postgres** with a specific port, e.g., 1234:

```
$ postgres -p 1234
```

To connect to this server using psql, specify this port with the -p option:

```
$ psql -p 1234
```

or set the environment variable **PGPORT**:

```
$ export PGPORT=1234
$ psql
```

Named run-time parameters can be set in either of these styles:

```
$ postgres -c work_mem=1234
$ postgres --work-mem=1234
```

Either form overrides whatever setting might exist for *work_mem* in postgresql.conf. Notice that underscores in parameter names can be written as either underscore or dash on the command line. Except for short-term experiments, it's probably better practice to edit the setting in postgresql.conf than to rely on a command-line switch to set a parameter.

SEE ALSO

initdb(1), **pg_ctl**(1)